AF /2123

IFW

PTO/SB/21 (09-04)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

OIPE
OCT 31 2005
PATENT & TRADEMARK OFFICE

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	09/879,658
Filing Date	June 11, 2001
First Named Inventor	Takahide Ohkami
Art Unit	2123
Examiner Name	A. Sharon
Attorney Docket Number	706316-1178
Total Number of Pages in This Submission	

ENCLOSURES (*Check all that apply*)

- [x] Fee Transmittal Form
  - [ ] Fee Attached
- [ ] Amendment/Reply
  - [ ] After Final
  - [ ] Affidavits/declaration(s)
- [ ] Extension of Time Request
- [ ] Express Abandonment Request
- [ ] Information Disclosure Statement
- [ ] Certified Copy of Priority Document(s)
- [ ] Reply to Missing Parts/ Incomplete Application
  - [ ] Reply to Missing Parts under 37 CFR 1.52 or 1.53
- [ ] Drawing(s)
- [ ] Licensing-related Papers
- [ ] Petition
- [ ] Petition to Convert to a Provisional Application
- [ ] Power of Attorney, Revocation Change of Correspondence Address
- [ ] Terminal Disclaimer
- [ ] Request for Refund
- [ ] CD, Number of CD(s) ______
  - [ ] Landscape Table on CD
- [ ] After Allowance Communication to TC
- [ ] Appeal Communication to Board of Appeals and Interferences
- [x] Appeal Communication to TC (**Appeal Notice, Brief, Reply Brief**)
- [ ] Proprietary Information
- [ ] Status Letter
- [ ] Other Enclosure(s) (please Identify below):

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm Name	Orrick, Herrington & Sutcliffe LLP		
Signature			
Printed name	Jeffrey A. Miller		
Date	10/28/05	Reg. No.	35,287

CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being facsimile transmitted to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below:

Signature			
Typed or printed name	Jeffrey A. Miller	Date	10/28/05

PTO/SB/17 (12-04v2)
Approved for use through 07/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

Effective on 12/08/2004.
Fees pursuant to the Consolidated Appropriations Act, 2005 (H.R. 4818).

FEE TRANSMITTAL
For FY 2005

☐ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT ($)

Complete if Known	
Application Number	09/879,658
Filing Date	June 11, 2001
First Named Inventor	Takahide Ohkami
Examiner Name	A. Sharon
Art Unit	2123
Attorney Docket No.	706316-1178

METHOD OF PAYMENT (check all that apply)

☐ Check ☐ Credit Card ☐ Money Order ☐ None ☐ Other (please identify):

☑ Deposit Account Deposit Account Number: 15-0665 Deposit Account Name: Orrick Herrington & Sutcliffe LLP

For the above-identified deposit account, the Director is hereby authorized to: (check all that apply)

☑ Charge fee(s) indicated below ☐ Charge fee(s) indicated below, **except for the filing fee**

☑ Charge any additional fee(s) or underpayments of fee(s) under 37 CFR 1.16 and 1.17 ☑ Credit any overpayments

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

FEE CALCULATION

1. BASIC FILING, SEARCH, AND EXAMINATION FEES

Application Type	FILING FEES Fee ($)	FILING FEES Small Entity Fee ($)	SEARCH FEES Fee ($)	SEARCH FEES Small Entity Fee ($)	EXAMINATION FEES Fee ($)	EXAMINATION FEES Small Entity Fee ($)	Fees Paid ($)
Utility	300	150	500	250	200	100	
Design	200	100	100	50	130	65	
Plant	200	100	300	150	160	80	
Reissue	300	150	500	250	600	300	
Provisional	200	100	0	0	0	0	

2. EXCESS CLAIM FEES

Fee Description	Fee ($)	Small Entity Fee ($)
Each claim over 20 (including Reissues)	50	25
Each independent claim over 3 (including Reissues)	200	100
Multiple dependent claims	360	180

Total Claims ____ - 20 or HP = **Extra Claims** ____ x **Fee ($)** ____ = **Fee Paid ($)** ____

HP = highest number of total claims paid for, if greater than 20.

Indep. Claims ____ - 3 or HP = **Extra Claims** ____ x **Fee ($)** ____ = **Fee Paid ($)** ____

HP = highest number of independent claims paid for, if greater than 3.

Multiple Dependent Claims — **Fee ($)** ____ **Fee Paid ($)** ____

3. APPLICATION SIZE FEE

If the specification and drawings exceed 100 sheets of paper (excluding electronically filed sequence or computer listings under 37 CFR 1.52(e)), the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).

Total Sheets ____ - 100 = **Extra Sheets** ____ / 50 = **Number of each additional 50 or fraction thereof** ____ (round **up** to a whole number) x **Fee ($)** ____ = **Fee Paid ($)** ____

4. OTHER FEE(S) **Fees Paid ($)**

Non-English Specification, $130 fee (no small entity discount) ____

Other (e.g., late filing surcharge): Appeal Brief $500

SUBMITTED BY

Signature		Registration No. (Attorney/Agent) 35,287	Telephone 650-614-7660
Name (Print/Type) Jeffrey A. Miller			Date 10/28/05

706316-1178
Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants: **Takahide Ohkami**	)	Group Art Unit 2123
	)	
Serial No. 09/879,658	)	Examiner A. Sharon
	)	
Filed: June 11, 2001	)	October 28, 2005
	)	
For: HARDWARE-ASSISTED DESIGN VERIFICATION SYSTEM USING A PACKET-BASED PROTOCOL LOGIC SYNTHESIZED FOR EFFICIENT DATA LOADING AND UNLOADING	)	

APPEAL BRIEF

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This is an appeal to the Board of Patent Appeals and Interferences from a Final Office Action dated April 15, 2005 and a subsequent Advisory Action dated August 4, 2005. A Notice of Appeal was timely submitted on September 16, 2005.

CERTIFICATE OF MAILING
(37 C.F.R. § 1.8(a))

I have reasonable basis to expect that this paper (along with any referred to as being attached or enclosed) is being deposited with the United States Postal Service on the date shown below with sufficient postage as First Class Mail in an envelope addressed to the Mail Stop **Appeal Brief-Patents**, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Date of Deposit: October 28, 2005

Jeffrey A. Miller

The undersigned authorizes a charge to Deposit Account No. 15-0665 in the amount of $500.00 for the filing of this Appeal Brief (pursuant to 37 C.F.R. § 41.20(b)(2)) in the above-identified matter. The undersigned also authorizes any additional fees which may be required, or credit any overpayment to Deposit Account No. 15-0665.

Applicant submits this Appeal Brief in accordance with 37 C.F.R. § 1.192.

I. REAL PARTY IN INTEREST

The real party in interest is QUICKTURN DESIGN SYSTEMS, INC., a corporation organized and existing under and by virtue of the laws of the STATE OF DELAWARE and having its principal place of business at 2655 SEELY AVENUE, SAN JOSE, CA 95134. QUICKTURN DESIGN SYSTEMS, INC., is a wholly-owned subsidiary of CADENCE DESIGN SYSTEMS, INC., 2655 SEELY AVENUE, SAN JOSE, CA 95134.

II. RELATED APPEALS AND INTERFERENCES

NONE.

III. STATUS OF CLAIMS

Claims 1-7 are appealed. Each of these claims stands rejected. Claims 8 and 9 were cancelled during prosecution. Claim 10 has already been allowed.

IV. STATUS OF AMENDMENTS

After final rejection, Applicant amended claim 1 to correct a typographical error. This amendment was entered by the Examiner in the Advisory Action mailed August 9, 2005.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The claimed invention relates to a method and apparatus for hardware-assisted design verification. A hardware-assisted design verification system uses a specialized hardware machine that communicates with a host workstation through a special communication channel, dedicated specifically for signal visibility operations. In typical hardware-assisted design verification systems, the loading and unloading of data to and from the hardware device (e.g., a hardware accelerator or emulator) takes significant time, thus degrading the efficiency of the system. To minimize the time required for loading and unloading data between a host workstation and the hardware device, the presently claimed invention supplements the design being verified (referred to as the "design under verification", or "DUV") with specialized logic, referred to as accessibility logic. Certain additional claims, e.g., claim 6, require use of a packet-based protocol to perform data transfer operations.

Independent claim 1 is best understood with reference to Figure 19, which is a flow chart illustrative of method steps recited by claim 1. A netlist description of the user's design identifies all memories and registers in the target design 200. Then, accessibility logic is synthesized into the user's logic design 202, which creates access ports to the memories and registers in the user's design 204. As recited in claim 1, the accessibility logic supplements the user's logic. These access ports facilitate the reading and writing of data to and from the registers and memories in the target design.

Dependent Claims 2-5 recite additional limitations to independent claim 1. Claim 2 teaches the additional step of identifying each of the memories and registers in the user's design

with a unique identifier. Claim 3 recites a selecting logic that is added to the accessibility logic and is used to receive the unique identifier and select a particular memory or register. Claim 4 includes logic to read from or write to particular memories or registers. Finally, Claim 5 incorporates decode logic that receives commands from a host workstation and controls the reading and writing of data to particular memories and registers.

Independent Claim 6 recites an apparatus for verifying a target design by loading data to or unloading data from the registers and memories of that design. Protocol logic is synthesized into the user's design, and comprises an incoming packet register (e.g., reference numeral 80 of Fig. 3), an outgoing packet register (e.g., reference numeral 84 of Fig. 3), command decode logic (e.g., reference numeral 43 of Fig. 1), write command execution logic (e.g., reference numeral 44 of Fig. 1), read command execution logic(e.g., reference numeral 44 of Fig. 1), and interface logic. (e.g., reference numerals 61, 62, 64, 65, 67, 68, 71, 72, 73, 75, 76, 77 of Figs. 2A and 2B) The incoming and outgoing packet registers store packets being sent from and to the workstation. The command decode logic decodes the incoming packet register. The write command execution logic writes the data from the incoming packet register into the desired memory or register. Likewise, the read command execution logic reads data from the desired memory or register and stores that data in the outgoing packet register. Finally, the interface logic interfaces the registers and memories in the target design.

Dependent Claim 7 recites providing logic to perform two additional functions: a) determine whether the incoming packet register is new, and b) control the command decoding process.

The invention recited in the claims on appeal increases the bandwidth of the communication channel between the hardware-based functional verification system and the host workstation. This alleviates the severe degradation caused by the lengthy period of time required to write data into registers and memories and the equally problematic time period required to transfer data stored in registers and memories from the hardware-based verification system to the workstation for further analysis.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Whether claims 1-7 are unpatentable under 35 U.S.C. § 102 over Sample et al. (U.S. Patent No. 5,841,967) ("Sample")

VII. ARGUMENT

A. The Claims Are Not Anticipated By Sample.

Applicant respectfully traverses the rejection of claims 1-7 and respectfully requests that this appeal be sustained in its entirety. Applicant initially notes that it is a fundamental aspect of patent law that a reference cannot anticipate a claim unless that reference contains each and every limitation of the claim. As will be seen herein, each of the claims contain multiple limitations that are not disclosed by Sample.

B. Claim 1 is Not Anticipated By Sample.

1. Sample is not directed towards providing greater access to memories and registers within the user's design

As an initial matter, Applicant respectfully points out that the appealed claims are aimed at achieving a completely different result than Sample. Sample seeks to run simulation and emulation on a user's design simultaneously and efficiently. Typical verification systems running emulation and simulation concurrently run the simulator on a workstation, and the emulation portion on an emulator. Because the simulator and emulator are separate, time delays are introduced when events or signal state changes must be communicated between them. Sample reduces this time delay by moving the hardware running the simulator (e.g., processing devices) from the workstation to the emulator. See e.g., Sample, Col. 4, line 59, through Col. 5, line 4.

In contrast, the present claims seek to grant the host workstation much greater and more

efficient access to the user's design running on the hardware device. The claimed method and apparatus allow the host workstation to load data to and unload data from specified registers and memories at different points during emulation. As claim 1 states, this result is achieved by synthesizing accessibility logic into the user's design, which creates access ports to the desired registers and memories within the user's design. This logic is then synthesized into a user's design, and together they are loaded into the emulator. Unlike Sample, the presently claimed subject matter is not related to increasing efficiency while running a simulator and emulator concurrently; rather it is aimed at providing greater access to memories and registers in the user's design once that design is ultimately programmed into the hardware verification system. This fundamental difference makes it apparent that the present claims are distinguishable from Sample.

2. <u>Sample does not disclose anything regarding increasing communication channel bandwidth between a *host workstation* and another hardware device</u>

Sample discloses a method and apparatus for reducing the overhead needed to transfer data between an emulation system and a simulation module, where the simulation module is located *within* the emulator. Sample, Col. 3, lines 45-50 and Col. 4, lines 59-63. In particular, Sample discloses a system where simulation modules 14, which include processing *devices* 16, are connected through a programmable interconnect 12 so that they communicate with FPGAs 10. The FPGAs 10 emulate the design while simulation modules 14 simulate portions of the user's design that are described in behavioral formats. Sample, Col. 5, lines 23-32.

In contrast, the claims being appealed recite a method and apparatus that increase the bandwidth between a *host workstation* and another hardware device, namely a *functional verification system* (claim 1) or a hardware accelerator. (claim 6) Succinctly stated, Sample is directed to communication between two modules (emulation and simulation) *within* a single functional verification system while the present claims recite a method and apparatus for increasing communication bandwidth *between* a host workstation and either a functional verification system (claim 1) or a hardware accelerator. (claim 6)

That Sample is not at all concerned with communication between functional verification systems and host workstations is seen by the fact that it distinguishes what is taught therein from other systems that require communication between the host workstation and the hardware logic emulation system described therein. As seen at Col. 3, lines 28-43, Sample describes a system where there is communication *between* a host workstation and a hardware logic emulation system:

> One approach to combining emulation and simulation is to run a simulator on a host workstation (or network thereof) communicating the events or changes in signal state to and from the emulated portion of the design over a network interface. However, in such a solution, the speed of event transfer seriously limits performance. Experiments show that the average time to transfer a 4-byte data packet over transport control protocol ("TCP") running on a SUN workstation computer (e.g., a SPARC-20) is around 50 microseconds. Assuming that a data packet of such size is used for encoding an event and given average design activity of 1000 events per simulation cycle, the speed of simulation will be limited to 20 cycles per second. *Therefore, there currently exists a need for combining emulation and simulation to efficiently verify circuit designs that may be a mixture of gate-level, structural and behavioral representations.* [Italics added]

As seen from this quote from Sample, one approach to running both emulation (which is hardware based) and simulation (which is software based) on a user's design is to run the simulation in the host workstation, run the emulation in the emulator, and transfer data back and forth between the two systems. Sample, however, notes that "in such a solution, the speed of event transfer seriously limits performance."

Sample overcomes this problem by eliminating the need to transfer data back and forth between the emulator and the host workstation. Instead, Sample teaches moving the simulator *from* the workstation *to* the hardware-assisted verification system (i.e., the emulator). See e.g., Col. 4, line 59, through Col. 5, line 4:

> FIG. 1 shows the preferred embodiment of the logic verification system. The system includes one or more reconfigurable logic components which may be programmable gate array ("FPGA") devices 10 interconnected using the programmable interconnect 12. The interconnect 12 can be programmed to create an arbitrary connection between any number of inputs or outputs of the devices connected to it. The apparatus also includes one or more simulation modules 14 (for exemplary purposes only, three are shown). Each of the simulation modules 14 includes a microprocessor 16, connected through a microprocessor bus 18 to one or more random access memory devices 20, one or more reconfigurable logic components which may be FPGAs 22, and a system bus controller 24.

Thus, Sample discloses that the simulation modules 14 disclosed therein are part of the hardware logic emulation system. This teaching is clear, in that Sample states "*[t]he apparatus* [i.e., system with the FPGAs] also includes one or more simulation modules 14..." Such a system eliminates any need for the emulator to transfer data back and forth to a simulator running

on the host workstation.

In complete contrast to Sample, the present claims are directed to a system that increases communication bandwidth *between* the hardware assisted functional verification system (e.g., the emulator) *and* the workstation. Thus, the present claims require exactly what Sample discloses you should not do, namely communicate *between* the hardware assisted functional verification system *and* the host workstation.

The subject matter of the claims on appeal all contain limitations requiring that the communication bandwidth be increased *between the host workstation and the hardware-based functional verification system*. Claim 1 states:

> A method for compiling a user's logic design for implementation into a functional verification system such that communication bandwidth is increased *between the functional verification system and a host workstation*

Similarly, claim 6 states:

> ...said verification system having a host workstation in communication with a hardware accelerator...

Thus, each of the claims on appeal contain limitations that are completely absent from Sample, namely communication between a host workstation and a separate structure.

3. Sample Does Not Disclose "Accessibility Logic"

Claim 1 requires that "accessibility logic" be synthesized into the user's design. Claim 1 further requires that this accessibility logic supplement the user's logic. Claim 1 also requires that the accessibility logic create "access ports to the memories and registers in the user's logic

design, the access ports facilitating writing of data received from the host workstation to the memories." The Office Action and Advisory Action have both erroneously stated that Sample's "programmable interconnect" corresponds to the "accessibility logic" required by claim 1. Sample discloses that the programmable interconnect in his system is as follows:

> The system includes one or more reconfigurable logic components which may be programmable gate array ("FPGA") devices 10 interconnected using the programmable interconnect 12. The interconnect 12 can be programmed to create an arbitrary connection between any number of inputs or outputs of the devices connected to it.

Sample, Col. 4, lines 60-65. This quote from Sample makes it clear that Sample's programmable interconnect is different than the accessibility logic required by claim 1. Sample's "programmable interconnect" is used to arbitrarily interconnect the inputs and outputs of *the FPGA devices*. In contrast, the "accessibility logic" required by claim 1 is used to *create* access ports to the memories and registers *in the user's logic design.*[1] Applicant respectfully submits that physically interconnecting pins on a physical device (i.e., the FPGAs) is not the same thing as creating access ports to memories and registers that are in the form of a *netlist description* of a circuit design.[2]

---

[1] As the specification of the present application makes clear, the "user's logic design" is the design that will be verified in the functional verification system. See, e.g., Paragraph 19 of the present application.

[2] Applicant also notes that the claims do not attribute any interconnect function to the accessibility logic.

This significant distinction was brought to the Examiner's attention in the response Applicant filed on July 13, 2005. In the Advisory Action responding to this argument, the Examiner states:

> FPGA devices are Field Programmable Gate Arrays ("FPGA"). It is the FPGA that contains the user's logic design, and therefore, the connection of inputs and outputs to the FPGA inherently connects ports in the user's logic design.

As seen from the above-quote, the Advisory Action states that "ports" are part of the user's design. This statement ignores the language of claim 1, which requires that the accessibility logic "create access ports to the memories and registers *in the user's logic design* (italics added)." Thus, claim 1 contains a limitation, i.e., that the accessibility logic *creates* access ports *in the user's design*, that is simply missing from Sample. The Examiner's misreading of the claim language implicitly recognizes this.

4. Sample does not disclose "synthesis" of accessibility logic into a user's design

In addition, the Examiner incorrectly argues that Sample discloses *synthesis* of accessibility logic into the user's design. In the Advisory Action, the Examiner states that accessibility logic is inherently synthesized into the user's design because "otherwise Sample's programmable interconnect ... would have nothing to connect to."[3] This is not correct. All integrated circuit designs have interconnect, which is used to interconnect the various circuit elements. The programmable interconnect in Sample is no different than the interconnect on any

[3] This particular inherency argument was raised by the Examiner for the first time in the Advisory Action.

integrated circuit design, with one very significant difference. The interconnect in Sample must be "programmable" so that the interconnect can change to accommodate an infinite number of integrated circuit designs. Moreover, Examiner's argument is also wrong because the programmable interconnect of Sample is a hardware feature of the emulator. See Sample, Col. 4, lines 59-65. Thus, the programmable interconnect of Sample cannot possibly be synthesized into the user's design, as the user's design is not a hardware feature of the functional verification system.

As is seen in the Advisory Action, the Examiner shifted his rejection on this claim feature to rely on inherency. In the April 22, Office Action, no argument regarding inherency was made. Regardless, in order to support a rejection based on inherency, "the examiner must provide a basis in fact and/or technical reasoning to reasonably support the determination that the allegedly inherent characteristic <u>necessarily</u> flows from the teachings of the applied prior art." MPEP 2112. Here, the Examiner has completely failed to meet this burden. As discussed, the Examiner incorrectly equates the programmable interconnect of Sample, which is unquestionably a hardware feature of the emulator described by Sample, and the accessibility logic that is synthesized into the user's design.

5. <u>Sample's programmable "interconnect" performs no "logic" functions and therefore cannot satisfy the accessibility "logic" limitation</u>

Claim 1 requires that accessibility *logic* be synthesized into the user's design. The Office Action and the Advisory Action both rely on the programmable interconnect 12 of Sample to satisfy the "accessibility logic" limitation of claim 1. However, Sample does not say anything

about the programmable interconnect 12 using any *logic* to perform their interconnect functions. In fact, while the claims of Sample say that FPGAs can be used as interconnect devices, Sample says absolutely nothing about using any of the logic resources on those FPGAs to perform their interconnection function. Regardless, even if the logic resources on the FPGA were somehow used for interconnect purposes, that logic would not be *synthesized into the user's design*, as is required by claim 1.

6. The Examiner incorrectly equates behavioral fragments discussed in Sample with the registers and memories of "user's design"

The Examiner is wrong when equating behavioral fragments described in Sample with the memories and registers of the "user's design." See April 22, 2005 Office Action at pages 4-5. In Sample, the behavioral fragments relate to behavioral representations of a design that will be simulated in the processor 16 of the simulation module 14, as selected by the FPGA. Sample, Col. 5, lines 26-29. In the claim language quoted by the Office Action (see page 4), the "user's design" is referred to in the context of the "accessibility logic" that will be *synthesized* into the user's design. Thus, equating behavioral fragments, which are not products of synthesis, with the accessibility logic is not correct.

7. The Examiner incorrectly equates Sample's system controller with the access ports of claim 1

The April 22, 2005 Office Action is also wrong when equating Sample's use of a system controller 26 that downloads configuration data into the FPGAs and executable data into the

random access memories with the access ports required by claim 1. Claim 1 requires that the accessibility logic, which creates access ports, facilitate writing data into and reading data from memories and register's in the user's design. The sections from Sample quoted by the Office Action make it clear that no such features are disclosed. First, the "system controller 26," as seen in Figure 1 of Sample, is external to the FPGAs and programmable interconnect. In fact, the system controller 26 is part of the Sample emulation hardware ("The system controller 26 is implemented using a commercial embedded controller board..." Col. 5, lines 62-64), and *is not synthesized into the user's design*, as is required by claim 1. Moreover, the configuration data downloaded into the FPGAs by the system controller 26 is in fact the data used to program the FPGAs. This configuration data is *not* data to be written into and read from memories and registers *in the user's design.*

Likewise, the system controller 26 downloads "executable data" into the random access memory devices 20 that form hardware components of Sample's emulator. This is not what is required by claim 1, which states that the access ports write data to and read data from the workstation into memories and registers *in the user's design.*

As is seen from this discussion, claim 1 contains several limitations not disclosed in Sample and is therefore should have been allowed. Moreover, claims 2-5 are dependent upon claim 1, meaning they also should have been allowed.

C. Claim 6 Is Not Anticipated By Sample.

Applicant respectfully submits that claim 6 contains numerous limitations that are not taught by Sample. Firstly, Applicant notes that many of the arguments set forth above with respect to Claim 1 apply equally to Claim 6. Additionally, Claim 6 requires that protocol logic be synthesized into the target logic design (i.e., the user's logic design). Applicant respectfully submits that Sample does not disclose synthesizing *protocol logic* into the target logic design. The April 22, 2005 Office Action points to Sample's discussion of token ring protocols as satisfying this limitation.[4] However, all Sample says is that the daisy chain 32 used to interconnect FPGAs is set up with a token ring protocol. This relates to the emulation hardware and has *nothing to do with the target logic design.* As discussed, Sample does not disclose anything about synthesizing any logic into the user's design where this logic supplements the user's logic.

Moreover, the April 22, 2005 Office Action incorrectly equates the random access memory devices 20 with the incoming and outgoing packet registers required by claim 6. Sample's random access memory devices 20 are physical memories that are part of the hardware used to construct the emulation system:

> Each of the simulation modules 14 includes a microprocessor 16, connected through a microprocessor bus 18 to one or more random access memory *devices* 20...

Sample, Col. 4, line 67 through Col. 5, line 3 (italics added). These random access memory *devices* are used by the emulation hardware to store executable data downloaded into them by

[4] The Advisory Action did not respond to any of the specific arguments Applicants made in its July 13, 2005 response.

system controller 26. In contrast, the packet registers of claim 6 are synthesized *into the target logic design* to store data and commands sent by the host workstation. The packet registers of claim 6 are *not* part of the functional verification system's hardware.

Likewise, the April 22, 2005 Office Action equates Sample's operation decoder 42 with claim 6's command decode logic. This is inconsistent with this Office Action's statement that the packet registers of claim 6 are taught by Sample's random access memories 20. Claim 6 requires that the command decode logic decode commands in the incoming packet register. However, as seen in Figs. 3 and 4 of Sample, the operation decoder 42 is not even connected to random access memory device 20, meaning that Sample's operation decoder 42 could not possibly decode commands since it will never receive them. The same is true regarding the Office Action's arguments regarding the write command execution logic and read command execution logic.

Finally, the April 22, 2005 Office Action's arguments equating the interface logic required by clam 6 with the FPGAs disclosed by Sample is also erroneous. Claim 6 requires that the interface logic interface the registers and memories in the target logic circuit. In contrast, all Sample teaches is that the FPGAs have the target logic circuit programmed therein. Sample does not teach anything about *synthesizing interface logic into the target logic circuit*, which is what is required by claim 6.

In sum, claim 6 should have been allowed over Sample. Likewise, since claim 7 is dependent upon claim 6, claim 7 should have been allowed as well.

D. Conclusion

As is seen from the above discussion, Claims 1 and 6 are distinguishable from Sample. Moreover, Claims 2-5 and 7 are dependent on Claims 1 and 6, and are therefore distinguishable as well. Thus, it is respectfully requested that the Board reverse the Examiner on the issue presented in this brief and withdraw the rejection of Claims 1-7.

Respectfully submitted,

ORRICK, HERRINGTON & SUTCLIFFE LLP

Dated: October 28, 2005

By: ______________________

Jeffrey A. Miller
Reg. No. 35,287

Four Park Plaza, Suite 1600
Irvine, California 92614-2558
(650) 614-7660

APPENDIX A- PENDING CLAIMS

1. A method for compiling a user's logic design for implementation into a functional verification system such that communication bandwidth is increased between the functional verification system and a host workstation by allowing for greater read and write access to memories and registers in the user's logic design, comprising;

identifying all of the memories and registers in the user's logic design;

synthesizing accessibility logic into the user's logic design such that the user's logic design is supplemented by said accessibility logic, said accessibility logic creating access ports to the memories and registers in the user's logic design, the access ports facilitating writing of data received from the host workstation to the memories and registers in the user's logic design, said access ports further facilitating reading of data stored in the memories and registers in the user's logic design for transfer to the host workstation.

2. The method of claim 1 further comprising the step of assigning a unique identifier to each of the memories and registers in the user's logic design.

3. The method of claim 2 wherein said accessibility logic comprises selecting logic, said selecting logic adapted to receive said unique identifier and select a particular one of the memories and registers in the user's logic design.

4. The method of claim 3 wherein said accessibility logic comprises logic to read from or write to said particular one of the memories and registers in the user's logic design.

5. The method of claim 4 wherein said accessibility logic comprises decode logic that receives commands from a host and controls execution of reading and writing data to the memories and registers in the user's logic design.

6. A hardware-assisted design verification system for verifying a target logic circuit design, said verification system having a host workstation in communication with a hardware accelerator, the target logic circuit design comprising Boolean logic gates, registers and memories, the host workstation loading data to or unloading data from the registers and memories, comprising:

protocol logic synthesized into the target logic circuit design, said protocol logic comprising:

an incoming packet register in communication with said host workstation, said incoming packet register storing packets that include data and commands communicated from the host workstation;

an outgoing packet register in communication with said host workstation, said outgoing packet register storing packets that include data to be communicated to the host workstation;

command decode logic, said command decode logic decoding said commands in said incoming packet register to identify a particular operation, register or memory location in said target logic circuit design, where said particular operation can comprise a write command and where said particular operation can also comprise a read command;

write command execution logic to write data stored in said incoming packet register into said register or memory location in said target logic circuit design for a write command decoded at said command decode logic;

read command execution logic to read data from said register or memory location in said target logic circuit design and store said data in said outgoing packet register for a read command decoded at said command decode logic; and

interface logic interfacing said registers and memories in said target logic circuit design.

7. The hardware-assisted design verification system of claim 6, wherein said protocol logic includes logic to determine whether data from said incoming packet register is new and control activation of command decoding and execution.

10. A method of synthesizing a packet-based protocol logic for providing access to registers and memories in a target logic circuit design when performing functional verification using a hardware accelerator, comprising:

determining fixed sizes of a request packet, said request packet comprising tag, command, and data end fields;

counting how many of the registers are present in the target logic circuit design;

counting how many of the memories are present in the target logic circuit design;

determining a maximum identification field size of said request packet;

determining a maximum number of data bits of the registers in the target logic circuit design;

determining a maximum number of data bits of the memories in the target logic circuit design;

determining a maximum number of address bits of the memories in the target logic circuit design;

determining a maximum number of bits to send the register data, memory data, and memory address to the target logic circuit design to determine data field size of said request packet

creating an incoming packet register coupled to an input data buffer in the hardware accelerator;

creating an outgoing packet register coupled to an output data buffer in the hardware accelerator;

creating a command decode block to decode a command in said incoming packet register;

creating an execution logic to execute a command decoded at said decode block; and

creating interface logic to access the registers and memories in said target logic circuit design.

creating a memory identification register to identify the memories in the target logic circuit design;

creating a memory address register to provide a current memory address for access;

incrementing said current memory after a memory read command or a memory write command is executed;

creating a finite state machine to indicate that the packet-based protocol logic is in either non-memory mode, continuous memory write mode, or continuous memory read mode; and creating a state transition control that selects said non-memory mode when said continuous memory operation ends, said state transition control further selecting said continuous memory write mode when said continuous memory write operation is initiated, said state transition control further selecting said continuous memory read mode when said continuous memory read operation is initiated.

APPENDIX B- EVIDENCE

NONE.

APPENDIX C- RELATED PROCEEDINGS

NONE.